UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/844,993	04/27/2001	Jacob Dreyband	033144-017	1373

30139 7590 12/06/2006

WILSON & HAM
2530 BERRYESSA ROAD
PMB: 348
SAN JOSE, CA 95132

EXAMINER	
CHANNAVAJJALA, SRIRAMA T	
ART UNIT	PAPER NUMBER
2166	

DATE MAILED: 12/06/2006

Please find below and/or attached an Office communication concerning this application or proceeding.

UNITED STATES PATENT AND TRADEMARK OFFICE

Commissioner for Patents
United States Patent and Trademark Office
P.O. Box 1450
Alexandria, VA 22313-1450
www.uspto.gov

BEFORE THE BOARD OF PATENT APPEALS AND INTERFERENCES

Application Number: 09/844,993
Filing Date: April 27, 2001
Appellant(s): DREYBAND ET AL.

James H. Ortega
For Appellant

SUPPLEMENTAL EXAMINER'S ANSWER

This is in response to the missing "Evidence" relied upon section, page 3, paragraph (8), mailed on 15 June 2006, Examiner hereby issuing "**SUPPLEMENTAL EXAMINER'S ANSWER**", that including **"Evidence Relied upon"** by the examiner in the rejection of claims under appeal

This is in response to the appeal brief filed on 4/21/2006 appealing from the Office action mailed

(1) Real Party in Interest

A statement at page 2 identifying by name the real party in interest is contained in the brief is correct.

(2) Related Appeals and Interferences

The examiner is not aware of any related appeals, interferences, or judicial proceedings which will directly affect or be directly affected by or have a bearing on the Board's decision in the pending appeal.

(3) Status of Claims

The statement of the status of claims contained in the brief at page 2 is correct.

(4) Status of Amendments After Final

No amendment after final has been filed.

(5) Summary of Claimed Subject Matter

The summary of claimed subject matter contained in the brief at page 3-8 is correct.

(6) Grounds of Rejection to be Reviewed on Appeal

The appellant's statement at page 8 of the grounds of rejection to be reviewed on appeal is correct.

(7) Claims Appendix

The copy of the appealed claims contained in the Appendix page 14-21 to the brief is correct.

(8) Evidence Relied Upon

2002/0133811	**Duftler et al.**	**09 2002**
6,083,276	**Davidson et al.**	**07 2000**

Grady et al. "UML for XML schema mapping specification, [12/08/99], 8 pages

(9) Grounds of Rejection

The following ground(s) of rejection are applicable to the appealed claims:

Claim Rejections - 35 USC § 103

The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

This application currently names joint inventors. In considering patentability of the claims under 35 U.S.C. 103(a), the examiner presumes that the subject matter of the various claims was commonly owned at the time any inventions covered therein were made absent any evidence to the contrary. Applicant is advised of the obligation under 37 CFR 1.56 to point out the inventor and invention dates of each claim that was not commonly owned at the time a later invention was made in order for the examiner to consider the applicability of 35 U.S.C. 103(c) and potential 35 U.S.C. 102(e), (f) or (g) prior art under 35 U.S.C. 103(a).

1. Claims 1-3, 9, 15-17, 23, 29-31 are rejected under 35 U.S.C. 103(a) as being unpatentable over Grady et al., UML for XML schema mapping specification, 12/08/99 in view of Duftler et al., [hereafter Duftler], US Pub.No. 2002/0133811 filed on 14 Dec. 2000.

2. As to Claims 1,15,29, Grady et al., teaches a system which including 'mapping a descriptive language including a data description having a structure complexity into an object oriented programming language [see Abstract, page 2, 1.1], Grady is directed to standard object oriented language schemas, more specifically UML for XML schema mapping as detailed in Abstract, further Grady also suggests for example object management group where UML has been established certain standards, as best understood by the examiner, descriptive language is to enhance future extensibility and reusability of information in any embedded system for example XML is one of the suitable tool as detailed in Abstract, further it is noted that Grady specifically suggested unified modeling language or UML is a standard object-oriented language that corresponds to object oriented programming language [see Abstract, line 1-3];

'receiving the data description' [page 2, item 2], Grady specifically directed to mapping data types in XML schema to classes, further Grady teaches data types semantics that are related to XML schema concept, see table in page 3;

'identifying a complex-type element in the data description' [page 3, item 1.4,page 6, item 1.8], identifying complex-type element is integral part in the XML document instances of Grady because firstly Grady is directed to XML schema,[see

page 6, item 1.8], secondly, Grady specifically teaches for example defining two different data type(s) as detailed in page 3, item 1.4, further it is noted that complex types in XML schemas are user defined data types that can include other elements or attributes, complex types can contain elements defined as either simple or complex, complex types can also include attributes and groups, whereas simple types can only contain facets [see page 6, item:1.8], As best understood by the examiner, complex types are defined using the complex type element and typically contain combination of element, attribute, and group declaration, as well as references to globally declared elements and groups, further a complex type can be thought of as a mini-schema that defines the valid structure and data contained within a specific element as detailed in page 6, item 1.8;

'creating an executable object oriented class corresponding to the identified complex-type element, wherein the class includes an internal static class wherein the internal static class corresponds to the structure complexity of the data description' [page 4, 1'5, page 6 example the XML schema], as best understood by the examiner, static class analysis is a kind of data flow analysis that computes a set of classes for each variable and expression in a method is integral part of object oriented language such as C++, further it is noted that descriptive language is not only enhance future extensibility but also reusability of classes and methods,

for example a simple **static class** is created as shown below and this is common knowledge object oriented environment.

```
[code]
#include <iostream>
class test {
static int counter;
public:
int getcount() { return counter;}
test();
};
int test::counter = 0;

test::test() {
counter++;
}

int main(void) {
test xyz, bar;

cout << xyz.getcount() << "\n";
}
[/code].
```

Although, creating an executable object oriented class is integral part of Grady's teaching because firstly, Grady is directed to UML or Unified Modeling Language is itself a standard object-oriented design language [see ABSTRACT], secondly, executable UML is the next layer of abstraction depends on profile of UML, also describes the data and the behavior, further executable UML doesn't make coding decisions, and make use of compiler to generate code, thirdly executable UML is a subset of UML proper as detailed above, on the other hand, to be useful, this subset of UML must have a way to specify actions at a higher level of abstraction than a programming language. There are many executable profiles that could be defined to

select a set of meaningful components and how they interact during execution. For example, a object can invoke methods of other objects, which in turn invoke more methods is part of Grady's Unified modeling language

It is however, noted that Grady does not specifically teach "object oriented class that is independently executable in any of a plurality of run-time environments', although Grady specifically suggests unified modeling language or UML is a standard object-oriented design language. On the other hand, Duftler disclosed "object oriented class that is independently executable in any of a plurality of run-time environments' [page 1, col 1-2, 0009-0014], Duftler teaches object oriented language for example Java specifically defining and implementing XML as detailed in 0011-1113. As best understood by the examiner, Duftler also specifically suggests implementing JavaBeans components using any scripting language or languages that corresponds to any of a plurality of object oriented languages executable in run-time environments.

It would have been obvious to one of the ordinary skill in the art at the time of applicant's invention to incorporate the teachings of Duftler et al. into UML for XML schema mapping specification of Grady because both Grady and Duftler are directed to XML schema, and both are directed to object oriented language [see Grady: Abstract; Duftler: abstract, page 1, col 1, 0005-0006] and are from same field of endeavor.

One of the ordinary skill in the art at the time of applicant's invention to incorporate the teachings of Duftler et al. into UML for XML schema mapping of Grady because that would have allowed users of Grady to define and implement object oriented language for example Java independently executable in a run-time because Java is executable in any platform, further bringing the advantages of JavaBean components automatically generated at run time [see page 1, 0006].

3. Claims 8,14,22,28,36 most of the limitations of this claim have been noted in the rejection of Claim 1 above. In addition, with respect to the claimed feature Duftler disclosed 'naming space with said internal static class to provide an implementation of said structure complexity' [see page 9, 0122-0123].

4. As to Claims 2,16,30 Grady teaches a system which including 'receiving the data description comprises receiving an XML Schema [see page 6, 1.8 XML schema]. As best understood by the examiner, the purpose of XML schema is to define the building blocks of an DML document, just like a data type definition, further it should be noted fundamental XML schema defines such as: elements that appear in a document, defines attributes that appear, defines which elements are child elements, defines the order of child elements, defines the number of child elements, defines whether an element is empty or can include text, defines data types for elements and attributes, defines default and fixed values for elements and attributes [see Grady: page 4, item 1.5]

5. As to Claims 3,17,31, the limitations of this claim have been noted in the rejection of above claim. In addition, Grady disclosed 'validating the data description ' [see Abstract, page 4 1.5 defining element type]. As best understood by the examiner, XML Schema provides powerful dedicated validation features for things like uniqueness, referential integrity, enumerations, complex types and the various data type facet as suggested by Grady, at page 3, item 1.3.

6. As to Claims 9 and 23, Grady teaches a system which including 'mapping a schema including a structural complexity into an executable object oriented programming language wherein the object oriented programming language provides a one to one correspondence between the structural complexity of the Schema and the functionality of the object oriented programming language' [see Abstract], Grady specifically directed to unified modeling language which is a standard object oriented design language that is used by the object management group, further XML schema is integrated for example developing an object model that represented in DML, describing relationships between XML and system to process it as detailed in page , introduction, Schema corresponds to Grady's XML schema as detailed in page 2, item 1.1.

Although Grady teaches, an executable object oriented language which is integral part of Grady's teaching because firstly, Grady is directed to UML or Unified Modeling Language is itself a standard object-oriented design language [see ABSTRACT], secondly, executable UML is the next layer of abstraction depends on

profile of UML, also describes the data and the behavior, further executable UML doesn't make coding decisions, and make use of compiler to generate code, thirdly executable UML is a subset of UML proper as detailed above, on the other hand, to be useful, this subset of UML must have a way to specify actions at a higher level of abstraction than a programming language. There are many executable profiles that could be defined to select a set of meaningful components and how they interact during execution. For example, a object can invoke methods of other objects, which in turn invoke more methods is part of Grady's Unified modeling language;

'receiving said schema' [page 3, item 3, 1.3, section 4, page 6], schema corresponds to XML schema as detailed in section 4, page 6,;

'validating said schema' [see Abstract, page 4 1.5 defining element type],

'creating a set of executable object oriented classes including a set of internal static classes to provide a mapping of the schema into the object oriented language' [page 4, 1'5, page 6 example the XML schema], as best understood by the examiner, static class analysis is a kind of data flow analysis that computes a set of classes for each variable and expression in a method is integral part of object oriented language such as C++, further it is noted that descriptive language is not only enhance future extensibility but also reusability of classes and methods:

for example a simple **static class** is created as shown below and this is common knowledge object oriented environment.

```
[code]
#include <iostream>
class test {
static int counter;
public:
int getcount() { return counter;}
test();
};
int test::counter = 0;

test::test() {
counter++;
}

int main(void) {
test xyz, bar;

cout << xyz.getcount() << "\n";
}
[/code].
```

It is however, noted that Grady does not specifically teach "independently executable in any of a plurality of run-time environments', although Grady specifically suggests unified modeling language or UML is a standard object-oriented design language. On the other hand, Duftler disclosed "object oriented class that is independently executable in any of a plurality of run-time environments' [page 1, col 1-2, 0009-0014], Duftler teaches object oriented language for example Java specifically defining and implementing XML as detailed in 0011-1113. As best understood by the examiner, Duftler also specifically suggests implementing JavaBeans components using

any scripting language or languages that correspond to any of a plurality of object oriented languages executable in run-time environments.

It would have been obvious to one of the ordinary skill in the art at the time of applicant's invention to incorporate the teachings of Duftler et al. into UML for XML schema mapping specification of Grady because both Grady and Duftler are directed to XML schema, and both are directed to object oriented language [see Grady: Abstract; Duftler: abstract, page 1, col 1, 0005-0006] and are from same field of endeavor.

One of the ordinary skill in the art at the time of applicant's invention to incorporate the teachings of Duftler et al. into UML for XML schema mapping of Grady because that would have allowed users of Grady to define and implement object oriented language for example Java independently executable in a run-time because Java is executable in any platform, further bringing the advantages of JavaBean components automatically generated at run time [see page 1, 0006].

7. Claims 4-14,18-28,32-36 are rejected under 35 U.S.C. 103(a) as being unpatentable over Grady et al., UML for XML schema mapping specification, 12/08/99, Duftler et al., [hereafter Dulftler], US 2002/0133811 as applied to claims 1,15,29 above, and further in view of Davidson et al., [hereafter Davidson], US Patent No. 6083276.

8. As to Claims 4,10,18,24,32, both Grady, Duftler teaches a system which including XML data description, mapping specification [Grady: see Abstract; Duftler: Abstract], however, Grady and Dulfler do not specifically teach 'mutator method', although both Grady, and Duftler suggests for example standard object oriented design language that is widely used in software development area [see Grady: Abstract; Duftler: Abstract]. On the other hand, Davidson disclosed 'mutator method' [col 24, line 65-67, col 25, line 1-7], examiner interpreting mutator method corresponds to Davidson's mutator methods as detailed in col 25, line 4-6, fig 5.

It would have been obvious to one of the ordinary skill in the art at the time of applicant's invention to incorporate the teachings of Davidson et al., into UML for XML schema mapping specification of Grady et al., and Bean scripting components which is XML based language for defining and implementing JavaBean of Duftler et al. because they all are directed to XML mapping the schema [see Grady et al., Abstract, page 2: 1.1; Duftler: Abstract,;Davidson: fig 1, element 122], they all are directed to descriptive language including a data description [see Grady et al. XML example page6; Duftler:

page 1, col 2, 0013; Davidson: col 8, line 10-20, col 21, line 50-65, fig 3A-4A] and they all are directed to XML environment and are both from the same field of endeavor.

One of ordinary skill in the art at the time of the invention would have been motivated to combine the references with Davidson et al. because that would have allowed users of Grady's UML for XML schema mapping, Bean scripting components which is XML based language for defining and implementing JavaBean of Duftler to control which relative combinations of specific properties, events, methods, classes satisfies his or her needs as suggested by Davidson et al [col 4, line 48-58].

9. As to Claims 5,11,19,25, and 33, both Grady and Davidson teach 'validity determination as to said data description' [see Grady: Abstract, page 2, 1.1; Davidson: fig 3A-4A], Davidson teaches 'sending request including said data description from a user to a remote server' [fig 1, col 7, line 30-40].

10. As to Claims 6,12,20,26,34, the limitations of this claim have been noted in the rejection of claim above. In addition, Davidson disclosed 'reading said data description into a set of valid descriptor classes' [col 9, line 41-52], 'creating a set of objects out of the data description wherein the occurrence of an object reflects validity' [col 10, line 4-20].

11. As to Claims 7,13,21,27,35, the limitations of this claim have been noted in the rejection of claim above. In addition, Davidson disclosed 'Java, C++,Smalltalk' [col 2, line 21-29].

(10) Response to Argument

a) At page 8, claims 1,9,15,23,29, applicant argues that "Unified modeling language (UML) discussed in Grady is not an executable object-oriented programming (OOP) language as recited in present independent claims 1,9,15,23,29. More specifically, while UML is an object oriented language, it is only a descriptive language and is not an independently executable programming language; hence, Grady's description of UML as an object oriented design language (i.e., a descriptive language), it is no more independently executable

b) At page 10, claims 1,9,15,23,29, applicant argues that the final rejection now cites Duftler for this missing element. In response, the appellants respectfully assert that the combination of Duftler with Grady still does not teach or suggest all of the elements of the pending independent claims

As to the argument [a-b], examiner disagrees with the applicant because firstly, Grady specifically suggests "UML or Unified modeling language" is a standard object-oriented design language [see Abstract, page 1, line 1-2], secondly, "UML" consists of several sublanguages which are "packaged in a suite" to model structural and behavioral aspects of a software system, thirdly, the main objective of "UML" is a

general purpose "object-oriented modeling language" instead of a domain-specific modeling language because "UML" as it is common in structured as well as in object-oriented modeling approaches for example commonly accepted notations, semantics as well as abstract syntax,

It is however, in the last office action, examiner clearly noted that although, creating an "*executable object oriented class* " is integral part of Grady's teaching because as stated reasons above, for example Grady is directed to UML or Unified Modeling Language is itself a standard object-oriented design language [see ABSTRACT], executable UML is the next layer of abstraction depends on profile of UML, also describes the data and the behavior, further executable UML doesn't make coding decisions, and make use of compiler to generate code, also, executable UML is a subset of UML proper as detailed above, on the other hand, to be useful, this subset of UML must have a way to specify actions at a higher level of abstraction than a programming language. There are many executable profiles that could be defined to select a set of meaningful components and how they interact during execution, for example, a object can invoke methods of other objects, which in turn invoke more methods is part of Grady's Unified modeling language.

It is however, noted that Grady does not specifically teach "object oriented class that is independently executable in any of a plurality of run-time environments', although Grady specifically suggests unified modeling language or UML is a standard object-oriented design language.

On the other hand, Duftler disclosed "object oriented class that is independently executable in any of a plurality of run-time environments' [page 1, col 1-2, 0009-0014], Duftler teaches object oriented language for example Java specifically defining and implementing XML as detailed in 0011-1113. As best understood by the examiner, Duftler also specifically suggests implementing JavaBeans components using any scripting language or languages that corresponds to any of a plurality of object oriented languages executable in run-time environments.

It would have been obvious to one of the ordinary skill in the art at the time of applicant's invention to incorporate the teachings of Duftler et al. into UML for XML schema mapping specification of Grady because both Grady and Duftler are directed to XML schema, and both are directed to object oriented language [see Grady: Abstract; Duftler: abstract, page 1, col 1, 0005-0006], both Grady, Duftler specifically teaches "object-oriented language" [see Grady: Abstract, line 1-2; Duftler: :Abstract, line 6-8, page 1, col 1, 0005, 0006, line 1-3], particularly, Duftler suggests Java platform and JavaBeans implemented using any programming language, and Grady, Duftler are at least based on "*object-oriented language*" and they both are from same field of endeavor..

One of the ordinary skill in the art at the time of applicant's invention to incorporate the teachings of Duftler et al. into UML for XML schema mapping of Grady because that would have allowed users of Grady to define and implement object oriented language for example Java independently executable in a run-time because Java is executable in any platform, further bringing the advantages of JavaBean components automatically generated at run time [see page 1, 0006].

c) At page 10, claims 1,9,15,23,29, applicant argues that "Duftler merely teaches a different means for creating JavaBeans to be executed in Java code such that the JavaBeans do not have to be individually encoded. However, such code of Duftler is still not *independently* executable in a run-time environment. Stated another way, the only executable language discussed in Duftler is Java, which the appellants have previously demonstrated is not independently executable in any of a plurality of run-time environments.

d) At page 11, claims 1,9,15,23,29 applicant argues that "since neither Grady nor Duftler teaches or suggests an independently executable object oriented language, as recited in claims 1,9,15,23,29, this cited combination of references does not render the pending claims obvious.

As to the above argument [c-d], as best understood by the examiner, firstly, Duftler is directed to defining and implementing JavaBeans using XML language [page 1, col 1, 0003], secondly, Duftler also suggested that JavaBean code automatically generated at run-time [page 1, col 1, 0006, line 10-11], thirdly, Duftler suggests Bean Scripting Component or BSC that combines the concepts that including

JavaBean programming and code fragments from BSC because Bean Scripting component is a XML based language [see page 1, col 2, 0011-0013]. It is further noted Duftler specifically suggested that defining and implementing JavaBeans components using any scripting language or languages. As noted, program code written in Java run in the Java environment is called "run-time environment", Duftler specifically teaches and supports "java runtime environment" [see Abstract, page 1, col 1, 0006, line 6-11], therefore, Duftler specifically teaches independently executable in a run-time environment.

e) At page 11-12, claims 1,9,15,23,29, applicant argues that "there is no motivation for one skilled in the art who is employing a Schema language, such as in Grady and in the present application, to combine the teachings in Duftler since Dulftler merely teaches employing XML (a non-schema language) in a specific descriptive function. As mentioned above, Duftler is teaching the use of XML to "better" describe an object when creating JavaBeans which would not be done with a Schema code such as that employed in Grady. There is nothing in Duftler that suggests abandoning the XML descriptive language technique taught therein to try to create JavaBeans with a Schema language.

In response to applicant's argument [e] that there is no suggestion to combine the references, the examiner recognizes that obviousness can only be established by combining or modifying the teachings of the prior art to produce the claimed invention where there is some teaching, suggestion, or motivation to do so found either in the

references themselves or in the knowledge generally available to one of ordinary skill in the art. See *In re Fine*, 837 F.2d 1071, 5 USPQ2d 1596 (Fed. Cir. 1988) and *In re Jones*, 958 F.2d 347, 21 USPQ2d 1941 (Fed. Cir. 1992).

In this case, Grady is directed to "UML for XML schema mapping specification", more specifically, describing the relationship between "unified modeling language" which is a standard object-oriented design language and XML schema i.e., schema language for object-oriented XML [see page 2, line 3-4], also, examiner notes that the objective of "UML" is a general purpose "object-oriented modeling language" instead of a domain-specific modeling language because "UML" as it is common in structured as well as in object-oriented modeling approaches for example commonly accepted notations, semantics as well as abstract syntax, furthermore, Grady also specifically suggests for example "XML schema" [see page 6, 1.8]. Duftler et al. is directed to defining, implementing JavaBeans components using XML based language, more specifically, JavaBean components can be defined and implemented using any programming language [see Abstract, page 1, col 1, 0006] created as "Bean Scripting Components". It is also noted that Duftler particularly teaches defining class, public methods, for JavaBeans components using scripting language i.e., Bean Scripting components provides syntax for describing JavaBean page 1, col 2, 0013].

It is noted that although, creating an "*executable object oriented class* " is integral part of Grady's teaching because as stated reasons above, for example Grady is directed to UML or Unified Modeling Language is itself a standard object-oriented design language [see ABSTRACT], executable UML is the next layer of

abstraction depends on profile of UML, also describes the data and the behavior, further executable UML doesn't make coding decisions, and make use of compiler to generate code, also, executable UML is a subset of UML proper as detailed above, on the other hand, to be useful, this subset of UML must have a way to specify actions at a higher level of abstraction than a programming language. There are many executable profiles that could be defined to select a set of meaningful components and how they interact during execution, for example, a object can invoke methods of other objects, which in turn invoke more methods is part of Grady's Unified modeling language

It is however, noted that Grady does not specifically teach "object oriented class that is independently executable in any of a plurality of run-time environments', although Grady specifically suggests unified modeling language or UML is a standard object-oriented design language.

On the other hand, Duftler disclosed "object oriented class that is independently executable in any of a plurality of run-time environments' [page 1, col 1-2, 0009-0014], Duftler teaches object oriented language for example Java specifically defining and implementing XML as detailed in 0011-1113. As best understood by the examiner, Duftler also specifically suggests implementing JavaBeans components using any scripting language or languages that corresponds to any of a plurality of object oriented languages executable in run-time environments.

It would have been obvious to one of the ordinary skill in the art at the time of applicant's invention to incorporate the teachings of Duftler et al. into UML for XML schema mapping specification of Grady because both Grady and Duftler are directed to

XML schema, and both are directed to object oriented language [see Grady: Abstract; Duftler: abstract, page 1, col 1, 0005-0006], and both Grady and Duftler are within the standards of "W3C" recommendations [see Duftler: page 2, col 1, 0029; Grady: page 1, Abstract, line 3-4, line 10-12] both Grady, Duftler specifically teaches "object-oriented language" [see Grady: Abstract, line 1-2; Duftler: :Abstract, line 6-8, page 1, col 1, 0005, 0006, line 1-3], particularly, Duftler suggests Java platform and JavaBeans implemented using any programming language, and Grady, Duftler are at least based on "*object-oriented language*" and they both are from same field of endeavor.

One of the ordinary skill in the art at the time of applicant's invention to incorporate the teachings of Duftler et al. into UML for XML schema mapping of Grady because that would have allowed users of Grady to define and implement object oriented language for example Java independently executable in a run-time because Java is executable in any platform, further bringing the advantages of JavaBean components automatically generated at run time [see page 1, 0006].

Therefore, applicant's remarks at pages 8-13 are deemed not to be persuasive, and claims 1-3,9,15-17,23,29-31 stand rejected under 35 USC 103 as being unpatentable over Grady et al. in view of Duftler et al.

f) At page 12, claims 4-7,10-13,18-21,24-27,32-35, have also been rejected as obvious over Grady in view of Duftler, and further in view of Davidson. However, as discussed above, the pending independent claims are not obvious over the combination

of Grady and Duftler. Davidson does nothing to cure these deficiencies in Grady and Duftler, as has not been cited or applied for this reason.

As to the above argument [f], applicant's remarks are merely conclusory statements, without any support, without addressing examiner's particular interpretation of the references nor the *combination* of references as presented in the previous office action, and without specifying claim[s] address the issues raised by examiner. Accordingly, examiner repeats the rejection as previously presented.

In the previous office action, examiner noted that Grady, Duftler do not teach "mutator method", although both Grady, Duftler suggests particularly, "object oriented language" is popularly used in software development [see Grady: Abstract; Duftler: Abstract]. On the other hand, Davidson suggested "mutator method" that corresponds to Davidson's mutator method col 25, line 4-6, fig 5.

It would have been obvious to one of the ordinary skill in the art at the time of applicant's invention to incorporate the teachings of Davidson et al., into UML for XML schema mapping specification of Grady et al., and Bean scripting components which is XML based language for defining and implementing JavaBean of Duftler et al. because they all are directed to XML mapping the schema [see Grady et al., Abstract, page 2: 1.1; Duftler: Abstract,;Davidson: fig 1, element 122], they all are directed to descriptive language including a data description [see Grady et al. XML example page6; Duftler: page 1, col 2, 0013; Davidson: col 8, line 10-20, col 21, line 50-65, fig 3A-4A] and they all are directed to XML environment and are both from the same field of endeavor.

One of ordinary skill in the art at the time of the invention would have been motivated to combine the references with Davidson et al. because that would have allowed users of Grady's UML for XML schema mapping, Bean scripting components which is XML based language for defining and implementing JavaBean of Duftler to control which relative combinations of specific properties, events, methods, classes satisfies his or her needs as suggested by Davidson et al [col 4, line 48-58].

(11) Related Proceeding(s) Appendix

No decision rendered by a court or the Board is identified by the examiner in the Related Appeals and Interferences section of this examiner's answer.

For the above reasons, it is believed that the rejections should be sustained.

Respectfully submitted,

Srirama Channavajjala

Conferees:

Alam, Hosain
SPE AU 2166

Vo, Tim
SPE ,AU2168

December 4, 2006